The Journal of Open Source Software

yieldplotlib: A unified library for exoplanet yield code visualizations

Corey Spohn [1*], **Sarah Steiger** [2*], **and Alex R. Howe** [1,3,4]

1 Goddard Space Flight Center, United States **2** Space Telescope Science Institute, United States **3** The Catholic University of America, United States **4** Center for Research and Exploration in Space Science and Technology, NASA/GSFC, United States * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

NASA's next flagship observatory, as recommended by the Astro2020 decadal survey, is the Habitable World's Observatory (HWO) which has the ambitious goal to "search for biosignatures from a robust number of about ~25 habitable zone planets and be a transformative facility for general astrophysics". The total number of habitable zone planets detected is referred to as the exo-Earth "yield" and estimating it will be critically important to HWO's success. As HWO is being developed and trade spaces are explored, yield codes such as the Altruistic Yield Optimizer (AYO) and EXOSIMS that can calculate the expected number of detected and characterized planets for a given mission architecture are essential. While these yield codes have the same goal, they can be complex and have major differences in their inputs and outputs which makes comparing results difficult. The need for a unified library for visualizing the inputs and outputs of these yield codes in a complete, descriptive, and accessible way has therefore also become apparent. To this end we have developed `yieldplotlib`, an open-source Python library to communicate the results of yield codes to the broader community and produce publication-quality plots. Currently, there are modules for analyzing AYO and EXOSIMS, but `yieldplotlib` is easily extensible and support for other yield codes can be easily added in the future.

Statement of need

Expected exoplanet yield is an important metric when evaluating the success of proposed flagship space observatory architectures such as those currently being considered for the Habitable Worlds Observatory (HWO; Feinberg et al. (2024)). As HWO is being developed and trade spaces are explored, yield codes such as the Altruistic Yield Optimizer (AYO; Stark et al. (2014)) and EXOSIMS (Delacroix et al., 2016) that can calculate the expected number of detected and characterized planets for a given mission architecture are essential. While these yield codes have the same goal, they are complex, written in different programming languages (AYO is written in IDL and EXOSIMS is written in Python), calculate yield with different methods, and have major differences in their inputs and outputs. The need for a unified library for visualizing these yield codes in a complete, descriptive, and accessible way has therefore also become apparent. This is non-trivial due to the differing methods, syntaxes, structures, and assumptions that each of these codes make.

Despite the challenges, when these values are interrogated directly, new insights are achieved. Some of these insights are highlighted in detail in Stark et al. (2025) where a comparison of just the internal exposure time calculations of AYO and EXOSIMS revealed sources of previously unknown discrepancy. `yieldplotlib` is a continuation of that initial work, but aimed instead at the higher level yield products (such as the total number of detected exo-Earths) which are of the most direct interest.

DRAFT

To visualize the inputs and outputs of AYO and EXOSIMS, `yieldplotlib` uses a custom loading and parsing structure that allows for the easy access of equivalent data across each of the codes. This allows `yieldplotlib` to communicate the results of yield codes to the broader community and produce publication-quality plots without the need to understand the complex underlying yield codes that were used to generate the data. Currently `yieldplotlib` contains modules for analyzing AYO and EXOSIMS, but is easily extensible and support for other yield codes can be easily added in the future.

Methods and Functionality

Parsing and Getting Values

`yieldplotlib` provides a loading system with a unified interface for accessing data from the yield codes. The system manages the complex and inconsistent file structures of the AYO and EXOSIMS inputs and outputs by organizing them into a hierarchical tree of nodes representing files and directories. This abstraction creates a consistent API that allows users to query data without needing to understand and parse the underlying data products. For collaboration purposes, the valid queries are managed in a Google Sheet where collaborators have linked the EXOSIMS and AYO keys to a universal key in `yieldplotlib`. This Google Sheet is processed into a key_map which is updated daily. Users can update the sheet, download it as a CSV file, and process it locally for development. An example excerpt from the CSV file can be found in Figure 1.

yieldplotlib key_map.csv sample

yieldplotlib name	description	EXOSIMS name	EXOSIMS file	EXOSIMS Class	AYO name	AYO file	AYO Class
star_dist	Distance to the star (in parsecs).	star_dist	reduce-star-target.csv	EXOSIMSCSVFile	dist (pc)	target_list.csv	AYOCSVFile
yield_earth	Yield of Earth-like exoplanet candidates	exoE_det_alt_mean	reduce-earth.csv	EXOSIMSCSVFile	exoEarth candidate yield	observations.csv	AYOCSVFile

Figure 1: Example portion of the `yieldplotlib` key map CSV file containing the mappings between AYO, EXOSIMS, and `yieldplotlib` parameters.

Once the yield packages are loaded and parsed, a getter can be called on the directory objects to return the corresponding value from the respective yield code, for example:

```
from yieldplotlib.load import AYODirectory, EXOSIMSDirectory

ayo = AYODirectory(Path("path/to/my/ayo_data"))
exosims = EXOSIMSDirectory(Path("path/to/my/exosims_data"))

ayo.get("yield_earth")
exosims.get("yield_earth")
```

Yield input packages (YIPs) specifying input coronagraph parameters can also be loaded and accessed using the same file node and directory structure. This allows users to access key coronagraph performance metrics that serve as critical inputs to these yield codes. In order to process the YIPs, `yieldplotlib` uses `yippy`[1] as a backend, though the user interface is identical to generating the AYO and EXOSIMS directories, for example:

```
from yieldplotlib.load import YIPDirectory
```

[1] github.com/CoreySpohn/yippy

```
yip = YIPDirectory(Path("path/to/my/yip_data"))
```

Plotting

Generic and Comparison Plots

`yieldplotlib` extends the commonly used Python plotting package `matplotlib` to take advantage of the wide variety of customization options that `matplotlib` offers, as well as the extensive knowledge base many users of `yieldplotlib` will have with that package. The `yieldplotlib` generic plots are used for single yield run visualizations and can make scatter plots, standard plots, and histograms. This extension is achieved through a function that runs when `yieldplotlib` is imported that automatically adds new plotting methods (prefixed with `ypl_`) to `matplotlib`'s `Axes` class, allowing users to directly call methods like `ax.ypl_plot()` and `ax.ypl_scatter()` on any `matplotlib` axes object.

Also provided are comparison plots that can handle multiple yield runs and automatically create multi-panel figures.

Plotting Scripts

`yieldplotlib` contains scripts for generating common plots used in yield code visualizations to provide instant usability for comparing AYO and EXOSIMS as motivated by the rapid pace of the ongoing architecture trade studies for HWO. This also serves to provide examples on how the package can be used for those who want to adapt the generic `yieldplotlib` parsing structure and plotting methods to generate their own bespoke visualizations.

Figure 2 and Figure 3 show two different examples of these types of yield outputs. Figure 2 shows the fraction of a star's habitable zone that can be sampled by during the lifetime of a mission known as the "habitable zone completeness" with the two yield codes in side-by-side axes and using the same color bar for ease of comparison. Figure 3 shows histograms of the total number of detected planets found as a function of planet type in this case just using EXOSIMS.

DRAFT

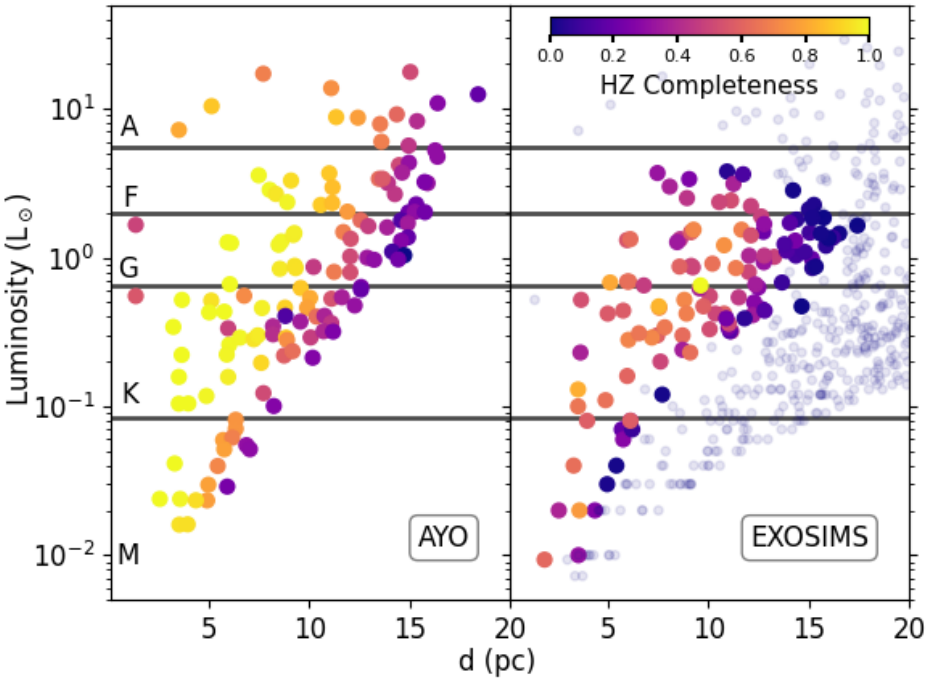

Figure 2: Plot of the Habitable Zone (HZ) completeness as a function of host star luminosity (in units of Solar luminosity) and distance (in parsecs). Here the AYO results are on the left and the EXOSIMS results are on the right.

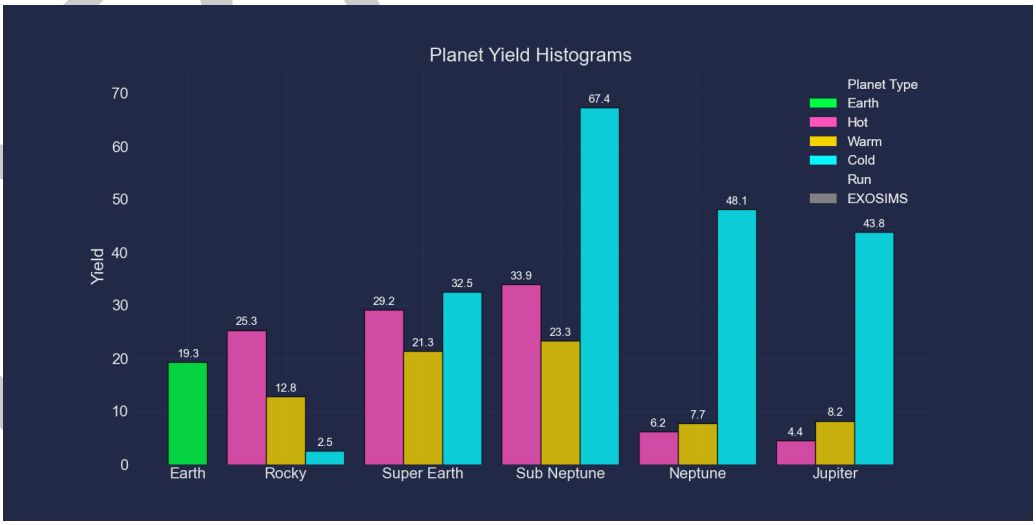

Figure 3: Bar chart showing expected EXOSIMS planet yields for hot (pink), warm (yellow), and cold (blue) Rocky planets, Super Earths, Sub-Neptunes, Neptunes and Jupiters. Earth-like planets which are of the most interest for HWO are shown in green. This plot uses the "cyberpunk" theme from the library `mplcyberpunk` which is supported as a keyword argument to `yieldplotlib` as a dark mode alternative to the standard plotting color schemes.

Yield code inputs can also have a profound impact on their results and so plotting these values is important to ensure consistency. Figure 4 shows the throughput for a key series of starlight

suppression optics in the observatory known collectively as a coronagraph. Smaller throughputs result in less planet light on the detector which can have a profound impact on final yields.

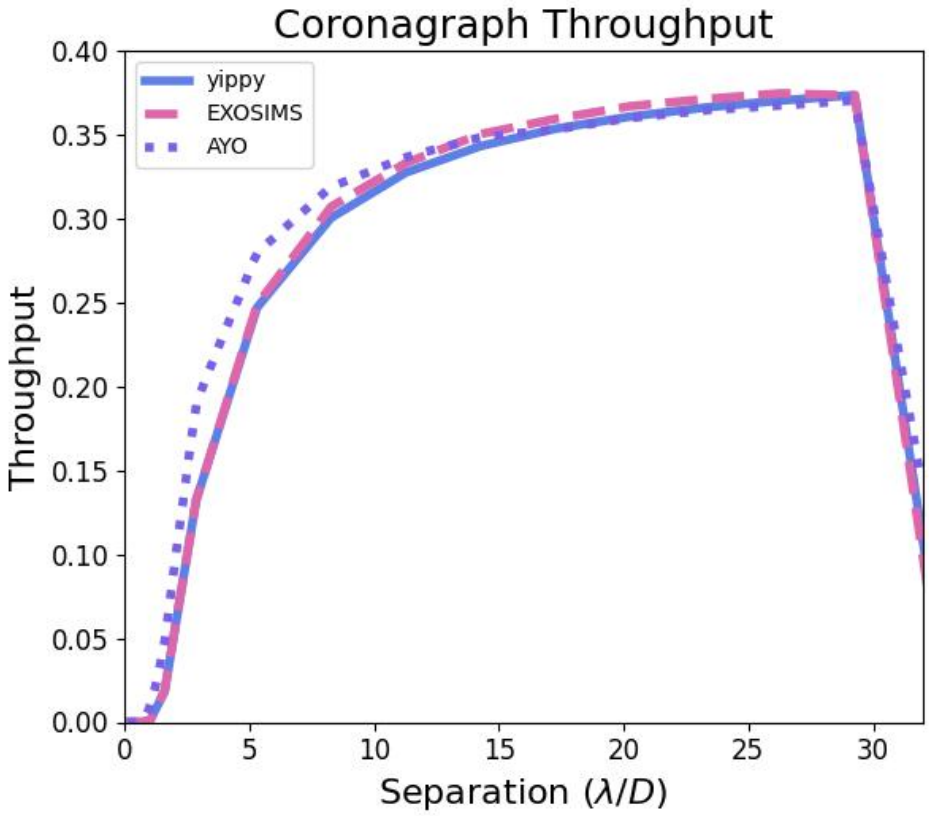

Figure 4: Core throughput vs. separation (in λ/D) for the coronagraph assumed by AYO (dotted purple), EXOSIMS (dashed pink), and pulled directly from the yield input package using yippy (solid blue). Slight differences between the codes can be attributed to how the "core" is defined. EXOSIMS and yippy adopt a fixed radius circular aperture whereas AYO defines an aperture based on pixels having more than 30% of the peak flux. Additional sources of difference can also lie in the interpolation methods used by all of the codes. This highlights the types of insights that tools like `yieldplotlib` can help to uncover.

DRAFT

Pipeline and Command Line Interface

In order to generate summary plots quickly, `yieldplotlib` comes packaged with a command line interface and plotting pipeline to create a suite of commonly used yield plots. This is accessed through the terminal by:

```
ypl_run path/to/yield/runs
```

where the specified path is either to a single folder containing the outputs for a single AYO or EXOSIMS run, or to a directory containing subdirectories of many AYO and EXOSIMS runs.

Future Work

Following up on the ETC cross-calibration work in Stark et al. (2025), a new cross-calibration study of high-level yield products is planned which will utilize `yieldplotlib`. This work will be vital to the ongoing HWO architecture trade studies by ensuring more reliable and robust

yield estimates, improving current yield codes, and providing a standard on which to calibrate new yield codes in the future.

Acknowledgements

Corey Spohn's research was supported by an appointment to the NASA Postdoctoral Program at the NASA Goddard Space Flight Center, administered by Oak Ridge Associated Universities under contract with NASA. Sarah Steiger acknowledges support from an STScI Postdoctoral Fellowship.

The authors would also like to acknowledge Christopher Stark, Dmitry Savransky, Rhonda Morgan, and Armen Tokadjian for providing consultation on the AYO and EXOSIMS repositories. They would also like to thank Justin Hom, and the rest of the Exoplanet Science Yields Working Group (ESYWG) for their valuable feedback and discussions as well as Susan Redmond for the development of the sample Yield Input Package.

Delacroix, C., Savransky, D., Garrett, D., Lowrance, P., & Morgan, R. (2016). Science yield modeling with the Exoplanet Open-Source Imaging Mission Simulator (EXOSIMS). In G. Z. Angeli & P. Dierickx (Eds.), *Modeling, systems engineering, and project management for astronomy VI* (Vol. 9911, p. 991119). https://doi.org/10.1117/12.2233913

Feinberg, L., Ziemer, J., Ansdell, M., Crooke, J., Dressing, C., Mennesson, B., O'Meara, J., Pepper, J., & Roberge, A. (2024). The Habitable Worlds Observatory engineering view: status, plans, and opportunities. In L. E. Coyle, S. Matsuura, & M. D. Perrin (Eds.), *Space telescopes and instrumentation 2024: Optical, infrared, and millimeter wave* (Vol. 13092, p. 130921N). International Society for Optics; Photonics; SPIE. https://doi.org/10.1117/12.3018328

Stark, C. C., Roberge, A., Mandell, A., & Robinson, T. D. (2014). Maximizing the ExoEarth Candidate Yield from a Future Direct Imaging Mission. *795*(2), 122. https://doi.org/10.1088/0004-637X/795/2/122

Stark, C. C., Steiger, S., Tokadjian, A., Savransky, D., Belikov, R., Chen, P., Krist, J., Macintosh, B., Morgan, R., Pueyo, L., Sirbu, D., & Stapelfeldt, K. (2025). Cross-Model Validation of Coronagraphic Exposure Time Calculators for the Habitable Worlds Observatory: A Report from the Exoplanet Science Yield sub-Working Group. *arXiv e-Prints*, arXiv:2502.18556. https://arxiv.org/abs/2502.18556

DRAFT